UCH UNIVERSIDAD DE CIENCIAS Y HUMANIDADES

INGENIERÍA DE SOFTWARE III

Eric Gustavo Coronel Castillo

gcoronelc.blogspot.com

ecoronel@uch.edu.pe

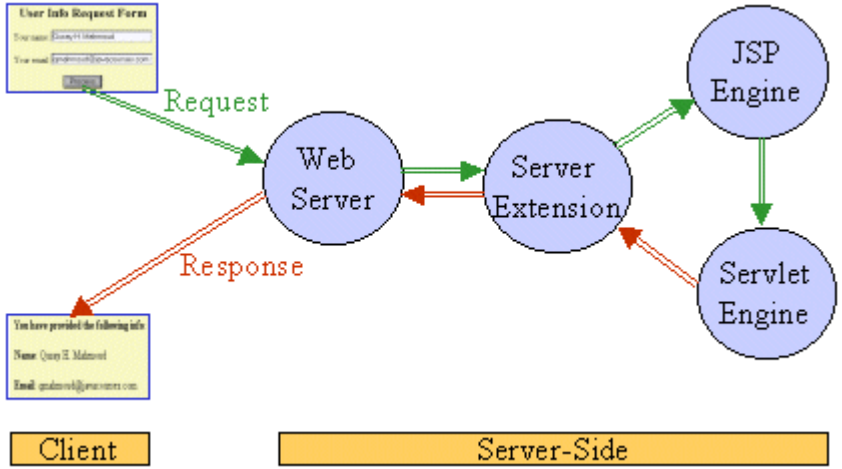
TEMA: Paginas JSP

Índice

- Objetivo
- ¿Qué es Java Server Pages?
- Elementos Básicos
- Directivas
- Acciones
- Objetos Implícitos

Objetivo

- Entender el funcionamiento de la tecnología Java Server Pages.
- Aplicar Java Server Pages en el desarrollo de aplicaciones web.



¿Qué es Java Server Pages?

- La tecnología Java Server Pages (JSP) aparece para aliviar lo difícil que supone generar páginas web mediante servlets.

- JSP es una tecnología que nos permite mezclar HTML estático con HTML generado dinámicamente.

- JSP es una especificación de Sun Microsystems. Sirve para crear y gestionar páginas Web dinámicas. Permite mezclar en una página código HTML para generar la parte estática, con contenido dinámico generado a partir de marcas especiales.

- El contenido dinámico se obtiene, en esencia, gracias a la posibilidad de incrustar dentro de la página código Java de diferentes formas. Su objetivo final es separar la interfaz (presentación visual) de la implementación (lógica de ejecución).

¿Qué es Java Server Pages?

Ejemplo

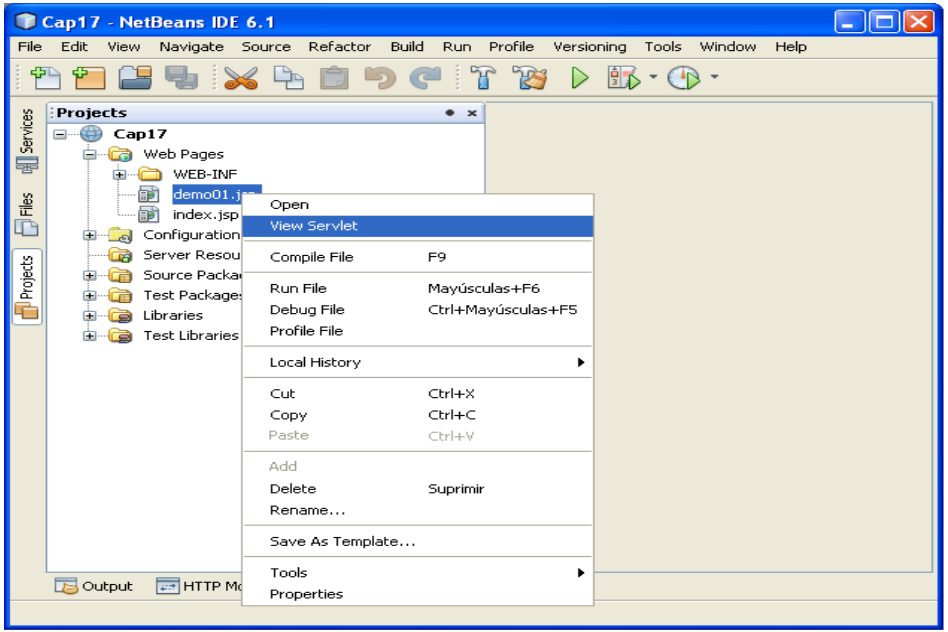
```
1.  <html>
2.  <head>
3.      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4.      <title>JSP - Demo 01</title>
5.  </head>
6.  <body>
7.      <%!
8.          private int n1 = 15;
9.          private int n2 = 10;
10.         private int s;
11.     %>
12.     <h4>Suma de Dos N&uacute;meros</h4>
13.     <%
14.         s = n1 + n2;
15.         out.println("n1 = " + n1 + "<br>");
16.         out.println("n2 = " + n2 + "<br>");
17.         out.println("suma = " + s + "<br>");
18.     %>
19. </body>
20. </html>
```

¿Qué es Java Server Pages?

- **Características**
  - La página JSP se convierte en un servlet.
  - La conversión la realiza el servidor o contenedor JEE, la primera vez que se solicita la página JSP.
  - Este servlet generado procesa cualquier requerimiento para esa página JSP.
  - Si se modifica el código de la página JSP, entonces se regenera y recompila automáticamente el servlet y se recarga la próxima vez que sea solicitada.

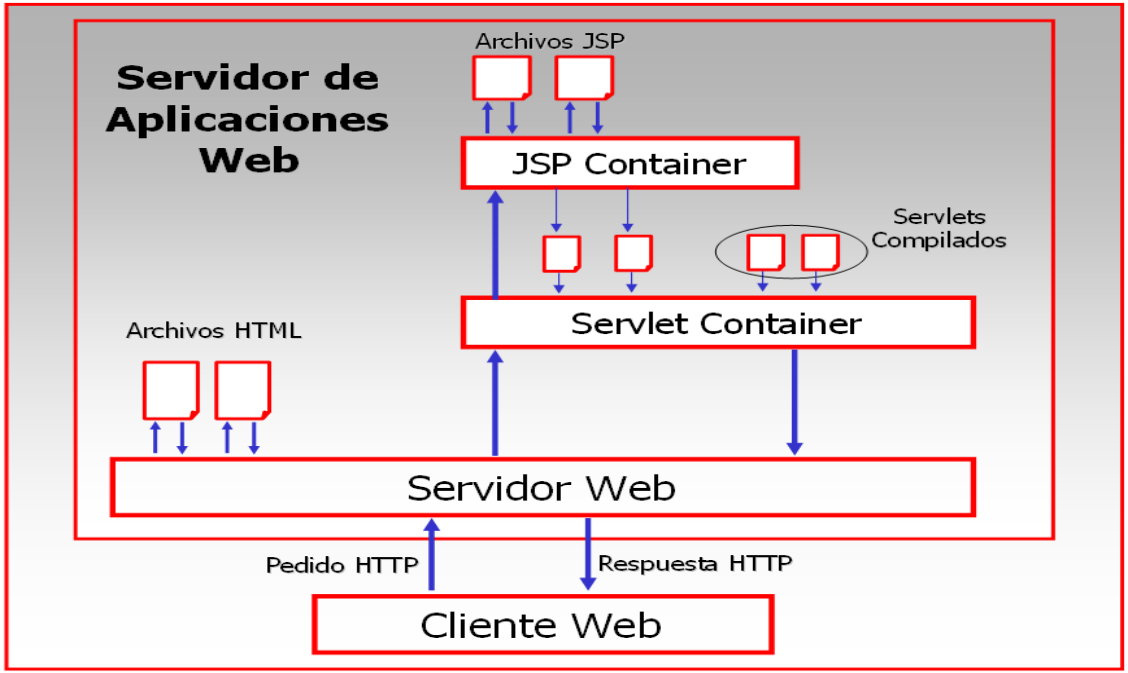
¿Qué es Java Server Pages?

- **Demostración**
  - Mostrar el Servlet que se genera a partir de una página JSP.



¿Qué es Java Server Pages?

- **Ciclo de Vida a Nivel de Documentos**

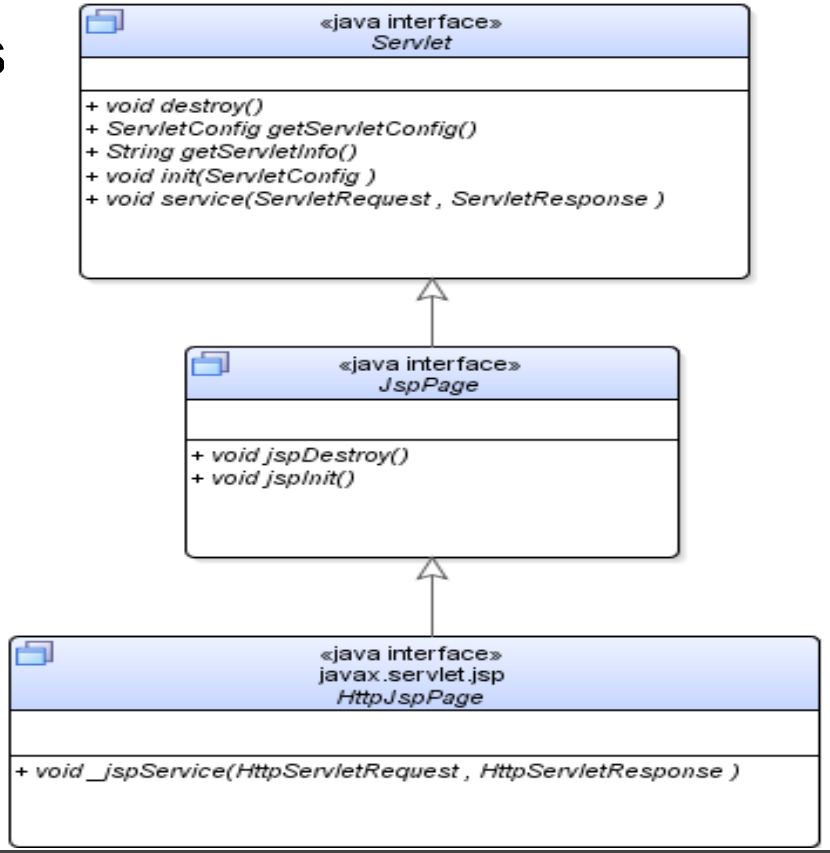


¿Qué es Java Server Pages?

- **Ciclo de Vida a Nivel de Eventos**

  Primero se ejecuta el método **jspInit()**, que podría ser programado, este método se ejecuta una sola vez, la primera vez que el jsp es requerido, análogamente al método init() de un servlet.

  La instancia creada para el servlet es también única, y es eliminada por el Garbage Collector del Servlet Engine, pero antes de eliminarse se ejecuta el método **jspDestroy()**, que podría ser también programado.



Elementos Básicos

- **Declaraciones**
  - **Sintaxis:**

```
<%!
        declaracion;
        [ declaracion; ]
        ...
%>
```

  - **Ejemplo:**

```
<%!
        private int cont = 0;
        private int num1, num2, num3;
        private Circulo objCirculo = new Circulo (2.0);
%>
```

Elementos Básicos

- **Expresiones**
  - **Sintaxis 1:**

    ```
    <%= expresión_Java %>
    ```

  - **Sintaxis 2:**

    ```
    <jsp:expression>
          Expresión Java
    </jsp:expression>
    ```

  - **Ejemplo:**

    ```
    Hora actual: <%= new java.util.Date() %>
    ```

Elementos Básicos

- **Scriptlets**
  - **Sintaxis 1:**

```
<%
        // Código Java
%>
```

  - **Sintaxis 2:**

```
<jsp:scriptlet>
        Código Java
</jsp:scriptlet>
```

  - **Ejemplo:**

```
if (Math.random() < 0.5) {
        out.write("Este es un <B>agradable</B> día!\n");
} else {
        out.write("Este es un <B>mal</B> día!\n");
}
```

Directivas

- **Introducción**

  – Una directiva JSP afecta a la estructura general de la clase servlet. A continuación tenemos la sintaxis:

```
<%@ directiva atributo="valor" %>
```

  – Sin embargo, también podemos combinar múltiples atributos para una sola directiva, a continuación tenemos la sintaxis:

```
<%@ directiva atributo1="valor1"
   atributo2="valor2"
   . . .
   atributoN="valorN" %>
```

Directivas

- **Introducción**

  – Hay tres tipos principales de directivas:

Directiva	Descripción
include	Se una para insertar un archivos dentro de la clase servlet, en el momento que la página JSP es traducida a un servlet
page	Se usa para definir atributos que se aplican toda la página jsp y cualquier archivo que se incluya con la directiva **include**.
taglib	Mediante esta directiva se puede ampliar el conjunto de etiquetas que el interprete de jsp es capaz de entender, de manera que la funcionalidad del mismo sea prácticamente ilimitada,

Directivas

- **Directiva: include**

  – Esta directiva nos permite incluir archivos en el momento en que la página JSP es traducida a un servlet.

  – Sintaxis:

```
<%@ include file="URL relativa" %>
```

Directivas

- **Directiva: page**
  - Define atributos que se aplican a una página JSP entera, y los archivos estáticos incluidos con la directiva include.
  - Sintaxis

    En **rojo** tenemos los valores por defecto.

```
<%@ page
    [ language="java"]
    [ extends="package .class"]
    [ import="{package .class | package.*}, ..." ]
    [ session="true|false"]
    [ buffer="none|8kb|sizekb"]
    [ autoFlush="true|false"]
    [ isThreadSafe="true|false"]
    [ info="text" ]
    [ errorPage="relativeURL"]
    [ contentType="{mimeType [;charset=characterSet ]" |
              text/html ; charset=ISO-8859-1}"]
    [ isErrorPage="true|false"]
%>
```

- **Directiva: taglib**

  – Indica al contenedor JEE que la página jsp utilizará librerías de etiquetas. Estas librerías contienen etiquetas creadas por el propio programador con sus correspondientes atributos que encapsulan determinada funcionalidad. Lo habitual es utilizar librerías públicas que han diseñado otros programadores y han sido profusamente probadas.

  – Sintaxis:

<%@ taglib uri=”uriLibreriaEtiquetas” prefix=”prefijoEtiqueta” %>

- **Directiva: taglib**
  - El contenido de cada atributo se describe a continuación:

Atributo	Descripción
uri	Permite localizar el archivo descriptor de la librería de etiquetas (TLD, Tag Library Descriptor).
prefix	Especifica el identificador que todas las etiquetas de la librería deben incorporar.

Acciones

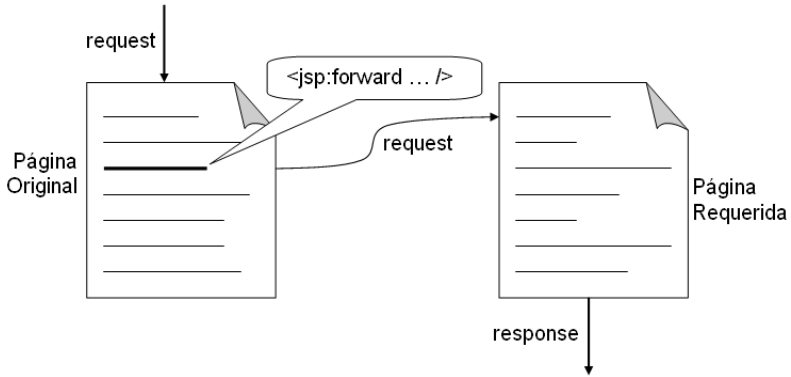
- Las acciones JSP usan sintaxis XML para controlar el comportamiento del motor de Servlets. Podemos insertar un fichero dinámicamente, reutilizar componentes JavaBeans, reenviar al usuario a otra página, o generar HTML.

- Hay que tener cuidado con las mayúsculas y las minúsculas ya que el motor es sensible (case sensitive) con respecto a como se escriben las etiquetas.

Acciones

- **Acción: <jsp:forward>**
  - Esta acción permite redirigir la ejecución de la página JSP actual hacia otro recurso de forma permanente, si antes de utilizar esta etiqueta ya se ha he enviado algún contenido del búfer del flujo de salida del cliente se producirá un error.
  - Sintaxis:

    <jsp:forward page="URLLocal"/>



Acciones

- **Acción: <jsp:param>**
  - Esta acción se utiliza en colaboración con cualquiera de las siguientes acciones: <jsp:forward>, <jsp:include> o <jsp:plugin>.
  - Sintaxis:

```
<jsp:param name=”nombreParametro” value=”valorParametro”/>
```

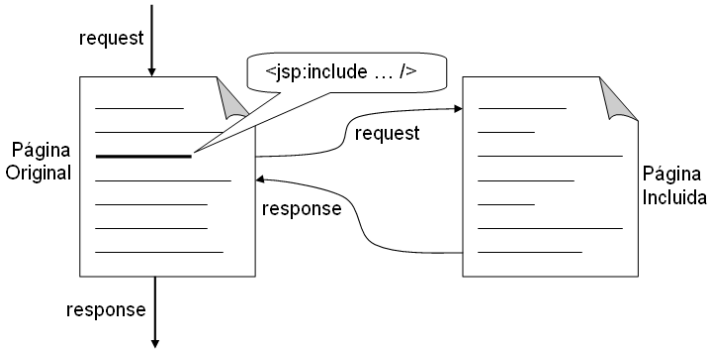
  - Ejemplo:

```
<jsp:forward page="demo.jsp">
    <jsp:param name="nombre" value="Gustavo"/>
    <jsp:param name="apellido" value="Coronel"/>
</jsp:forward>
```

Acciones

- **Acción: <jsp:include>**
  - La acción <jsp:include> permite insertar en el contenido generado por la página actual, el contenido de otro recurso distinto, resultando la salida final que se envía al usuario una combinación de ambos contenidos. Al contrario de lo que ocurría en la acción <jsp:forward>, el control de la ejecución vuelve a la página original una vez que se ha terminado la ejecución de la página incluida.
  - Sintaxis:

    <jsp:include page="URLLocal" flush="true|false"/>



- **Acción: <jsp:useBean>**

  - Esta acción se utiliza para poder utilizar dentro de una página JSP un componente JavaBean en un ámbito determinado. El componente JavaBean podrá ser utilizado dentro de la página JSP haciendo referencia al nombre indicado dentro de la acción <jsp:useBean>, teniendo siempre en cuenta el ámbito al que pertenece el Bean, y que se indica también en la acción <jsp:useBean>.

  - La acción <jsp:useBean> indica a la página JSP que queremos tener un Bean determinado disponible, el contenedor de páginas JSP creará el Bean correspondiente o bien lo recuperará del ámbito correspondiente.

Acciones

- **Acción: <jsp:useBean>**

Sintaxis:

```
<jsp:useBean
    id="beanInstanceName"
    scope="page | request | session | application"
    {
        class="package.class" |
        type="package.class" |
        class="package.class" type="package.class" |
        beanName="{package.class | <%= expression %>}“
                type="package.class"
    }
    {
        /> |
        > other elements </jsp:useBean>
    }
```

- **Acción: <jsp:useBean>**
  - Para localizar o instanciar el Bean, <jsp:useBean> sigue los siguientes pasos, en este orden:
    1. El contenedor trata de localizar un objeto que posea el identificador y ámbito indicados en la acción <jsp:useBean>.
    2. Si se encuentra el objeto, y se ha especificado un atributo type, el contenedor trata de utilizar la conversión de tipos del objeto encontrado con el tipo (type) especificado, si la conversión falla se lanzará una excepción ClassCastException. Si únicamente se ha indicado un atributo class se creará una nueva referencia al objeto en el ámbito indicado utilizando el identificador correspondiente.
    3. Si el objeto no se encuentra en el ámbito especificado y no se ha indicado un atributo class o beanName, se lanzará una excepción InstantiationException.

Acciones

- **Acción: <jsp:useBean>**

1. Si el objeto no se ha encontrado en el ámbito indicado, y se ha especificado una clase concreta con un constructor público sin argumentos, se instanciará un objeto de esa clase. Un nuevo objeto se asociará con el identificador y ámbitos indicados. Si no se da alguna de las condiciones comentadas, se lanzará una excepción InstantiationException.

2. Si el objeto no se localiza en el ámbito indicado, y se ha especificado un atributo beanName, se invocará al método instantiate(...) de la clase java.beans.Beans, pasándole como parámetro el valor de la propiedad beanName. Si el método tiene éxito un nuevo objeto se asociará con el identificador y ámbitos indicados.

3. Si la acción <jsp:useBean> no posee un cuerpo vacío, se procesará el cuerpo de la etiqueta, únicamente si el objeto se crea como un objeto nuevo, es decir, no se había encontrado en el ámbito indicado.

Acciones

- **Acción: <jsp:getProperty>**

  – Esta acción forma parte de las acciones que nos permiten utilizar componentes Beans dentro de nuestras páginas JSP, en este caso la acción <jsp:getProperty> nos va a permitir obtener el valor de la propiedad de un Bean creado en la página con el ámbito correspondiente.

  – La sintaxis de esta acción es muy sencilla, no posee cuerpo y únicamente presenta dos atributos, como se puede observar en la sintaxis.

  – Sintaxis:

```
<jsp:getProperty
    name=”nombreBean”
    property=”nombrePropiedad”/>
```

Acciones

- **Acción: <jsp:setProperty>**
  - Esta acción permite modificar las propiedades de los Beans a los que hacemos referencia en nuestras páginas JSP, es la acción complementaria a la acción <jsp.getProperty>.
  - Sintaxis:

```
<jsp:setProperty
     name=" nombreBean"
     {
          property= "*"    |
          property="nombrePropiedad" [
             param="nombreParametro" ]   |
          property="nombrePropiedad"
             value="{cadena | <%= expresión %>}"
     }
/>
```

- **Acción: <jsp:setProperty>**

  – El atributo **name** indica el identificador del Bean que hemos creado con la acción <jsp:useBean>.

  – Los detalles del atributo **property** son una serie de atributos que combinados entre sí permiten asignar el valor a la propiedad del Bean de distinta forma. Así por ejemplo la forma de establecer el valor de la propiedad de un Bean puede ser cualquiera de las que aparecen a continuación:

    - **property="*"**
    - **property="nombrePropiedad"**
    - **property="nombrePropiedad" param="nombreParámetro"**
    - **property="nombrePropiedad" value="valorPropiedad"**

- **Acción: <jsp:setProperty>**
  - El valor de una propiedad de un Bean se puede establecer a partir de varios elementos:
    - En el momento del requerimiento de la página a partir de los parámetros existentes en el objeto integrado request.
    - En el momento de ejecución de la página a partir de la evaluación de una expresión válida de JSP.
    - A partir de una cadena de caracteres indicada o como una constante en la propia página.

Objetos Implícitos

- Para simplificar el código en expresiones y scriplets, tenemos ocho variables definidas automáticamente, a estas variables también se les denomina "Objetos Implícitos".

- Estos objetos corresponden con objetos útiles del API de servlets (request, response, session, ...) y que en realidad son variables instanciadas de manera automática en el servlet generado a partir del JSP.

Objetos Implícitos

- **Objeto: request**
  - Este es el HttpServletRequest asociado con el requerimiento del cliente, y nos permite tener acceso a los parámetros asociados, para lo cual debemos utilizar el método **getParameter(..)**, el tipo de petición (GET, POST, HEAD, etc.), y las cabeceras HTTP entrantes (cookies, Referer, etc.).
  - Ejemplo:

```
<%
   String nombre = request.getParameter("nombre");
%>
```

Objetos Implícitos

- **Objeto: response**
  - Este es el HttpServletResponse asociado con la respuesta al cliente.
- **Objeto: out**
  - Este es el PrintWriter usado para enviar la salida al cliente. Sin embargo, para poder hacer útil el objeto response, esta es una versión con buffer de PrintWriter llamada JspWriter.
  - Ejemplo:

Mi Nombre: <% out.println(“Gustavo Coronel”); %>

Objetos Implícitos

- **Objeto: session**

  – Este es el objeto **HttpSession** asociado con el requerimiento. Recuerde que las sesiones se crean automáticamente, por lo tanto esta variable es creada incluso si no hubiera una sesión de referencia entrante.

  – La única excepción es cuando usamos el atributo **session** de la directiva **page** para desactivar las sesiones, en cuyo caso los intentos de referenciar la variable **session** causarán un error en el momento de traducir la página JSP a un servlet.

Objetos Implícitos

- **Objeto: session**
  - El objeto **session** también permite almacenar objetos en forma de atributos dentro del ámbito de la sesión, y por lo tanto presenta los mismos métodos que el objeto **request** para este propósito.
  - Ejemplo:

```
<%
   Empleado empleado = new egcc.dao.to.Empleado();
   empleado.setCodigo("12345");
   empleado.setNombre("Gustavo Coronel");
   session.setAttribute( "empleado", empleado );
%>
```

Objetos Implícitos

- **Objeto: application**

  – Este objeto integrado corresponde con el **ServletContext** obtenido mediante **getServletConfig().getContext()** y representa la aplicación Web a la que pertenece la página JSP actual.

  – El objeto **application** se encuentra disponible en cualquier página JSP y como instancia del interfaz **ServletContext** ofrece todos los métodos de esta interfaz, además de los métodos ya conocidos que se utilizan para almacenar y recuperar los atributos con ámbito de aplicación, no debemos olvidar que el objeto **application** define otro ámbito para los objetos que se almacenan como atributos, y en este caso es el ámbito más general posible.

Objetos Implícitos

- **Objeto: config**

  – Este objeto integrado es una instancia del interfaz **javax.servlet.ServletConfig**, y su función es la de almacenar información de configuración del servlet generado a partir de la página JSP correspondiente.

  – Normalmente las páginas JSP no suelen interactuar con parámetros de inicialización del servlet generado, por lo tanto el objeto **config** en la práctica no se suele utilizar.

Objetos Implícitos

- **Objeto: pageContext**

  – JSP presenta una nueva clase llamada **PageContext** para encapsular características de uso específicas del servidor.

  – Permite acceder al espacio de nombres de la página JSP actual, ofrece también acceso a varios atributos de la página así como una capa sobre los detalles de implementación.

  – Además el objeto **pageContext** ofrece una serie de métodos que permiten obtener el resto de los objetos integrados de JSP, también nos va a permitir acceder a los atributos pertenecientes a distintos ámbitos.

Objetos Implícitos

- **Objeto: pageContext**

  – JSP presenta una nueva clase llamada **PageContext** para encapsular características de uso específicas del servidor.

  – Permite acceder al espacio de nombres de la página JSP actual, ofrece también acceso a varios atributos de la página así como una capa sobre los detalles de implementación.

  – Además el objeto **pageContext** ofrece una serie de métodos que permiten obtener el resto de los objetos integrados de JSP, también nos va a permitir acceder a los atributos pertenecientes a distintos ámbitos.

Objetos Implícitos

- **Objeto: page**

  – El objeto **page** es una instancia de la clase **java.lang.Object**, y representa la página JSP actual, o para ser más exactos, una instancia de la clase del servlet generada a partir de la página JSP actual. Se puede utilizar para realizar una llamada a cualquiera de los métodos definidos por la clase del servlet.

  – Utilizar el objeto **page**, es similar a utilizar la referencia a la clase actual, es decir, la referencia **this**. Este objeto pertenece a la categoría de objetos integrados relacionados con los servlets, en esta caso representa una referencia al propio servlet.

  – En nuestras páginas JSP no es muy común utilizar el objeto **page**.

Proyecto 1

Desarrollar un proyecto que permita calcular el MCM y MCD de dos números enteros.

El usuario debe ingresar dos números enteros.

La aplicación debe mostrar el MCM y MCD.

Proyecto 2

- Desarrolle un proyecto que permita mostrar la tabla de multiplicar de un número.

- El usuario debe poder ingresar el número de la tabla que quiere mostrar.

Proyecto 3

- Desarrolle un proyecto que permita calcular el importe que se le debe pagar a un trabajador que labora por horas.

- Tener en cuenta que el trabajador entrega recibo por honorarios.

- Si el importe pasa los 1,500.00 Soles se le debe retener el 8% de renta.